

Appl. No. 09/666,629  
Request For Reconsideration dated August 4, 2004  
Reply to Final Office action of June 4, 2004

## REMARKS/ARGUMENTS

### Claims

Claims 1 through 11 were in this application and addressed by the Examiner in the present June 4, 2004 Office Action. The Examiner is thanked for withdrawing his previous rejections under section 112 as those rejections were evidently overcome by Applicants' previous March 5, 2004, filing. The present response does not further amend the pending claims but, instead, further analyzes the Examiner's rejections. Reconsideration and favorable action are respectfully requested.

### Rejections Under 35 U.S.C. § 103(a)

Claim 1 is rejected under 35 U.S.C. § 103(a) over Every in view of Ganssle and Kogure.

Looking to Kogure in detail, the Examiner states:

Kogure, however, teaches providing a program both static and dynamic allocation (Column 3, lines 54-60). Kogure also teaches a memory allocation and initiation system (Figure 3) for a plurality of frameworks in response to memory usage requirements (Figure 3, item 11) reported to the framework (Figure 3, items 15 and 16) by the algorithm module (Figure 3, item 10) through the memory interface (Figure 3, item 12).

Applicants, however, respectfully traverse this reading of Kogure as partially inaccurate, for reasons detailed below.

Kogure states that it is directed primarily to overcoming the inefficiency of memory swapping that occurs when memory is first used by one program in response to a static (i.e., beforehand) allocation, and then later a portion of that same memory is needed for use due to a static allocation of a second program – the inefficiency arises because there is required an interruption of the first program and the transfer, or “swapping,” of the contents of the memory that resulted from executing the first program to another storage area (e.g., external memory) so that the second program may properly operate.<sup>1</sup> The solution provided by Kogure combines a static and dynamic approach. However, this approach is not one that teaches or suggests the cited corresponding elements of claim 1, *with respect to the dynamic aspect*. To further appreciate this, the static aspect of Kogure is first discussed below, followed by a discussion of the dynamic aspect.

The static memory allocation aspect of Kogure is now explored. In Kogure, in static fashion, its “section 12 stores the information about an amount of memory necessary for initializing the program, which takes account of the amount of memory to

---

<sup>1</sup> Kogure, Col. 1, lines 17-24; 32-44.

Appl. No. 09/666,629

Request For Reconsideration dated August 4, 2004

Reply to Final Office action of June 4, 2004

be used by the program.”<sup>2</sup> Later, during processing, “memory management section 15 statically allocates the memory according to the notified memory amount.”<sup>3</sup> Thus, the “invention enables the loader 14 to request an allocation of a necessary amount memory to the memory management section 15 immediately before it loads a program, based on the program information stored in the management information section 12.”<sup>4</sup> In other words, and as discussed in connection with Kogure Figure 4, after some preliminary steps, “the loader 14 reads the information on the amount of memory required . . . [and] the loader 14 notifies the memory managements section 15” of that information.<sup>5</sup> The memory managements section 15 then actually allocates that memory based on that notification.<sup>6</sup> All of the preceding occurs in connection with the statically-determined memory requirements.

The *dynamic* memory allocation aspect of Kogure appears to be a safeguard when its static memory allocation aspect does not properly provide sufficient memory for an executing program. For example, Kogure states, “if the program executor 16 encounters a memory deficiency while it executes a program, the program executor 16 requests the memory management section 15 to allocate the deficiency in memory and the memory management section 15 dynamically allocates the requested memory.”<sup>7</sup> Curiously, though, Kogure nowhere appears to detail the manner in which its “memory management section 15” actually is informed as to what aspects of memory are needed to overcome the “deficiency.” In other words, the separate “executor 16” evidently detects when the statically-provided memory requirements are insufficient to actually support the executing program, but there is no indication of how the “memory management section 15” actually determines how much more (or type, etc.) of memory is needed to cure the deficiency. Indeed, Kogure appears to have a few different instances where it might be expected that its manner of doing so would be discussed, but the answer is not provided. For example, Kogure also states that “if a memory is deficient,” (meaning the memory allocation resulting from the static allocation), then “an additional memory is dynamically allocated”<sup>8</sup> – again, however, there is no statement as to how much or what type of a new allocation is provided to overcome the deficiency. As yet another example, Kogure Figure 8B illustrates the Kogure steps of detecting a program error when the statically provided memory is insufficient, but Kogure states in its text that upon such an error, “the amount of memory to be re-allocated is calculated from the amount of memory required at the occurrence of an error.”<sup>9</sup> However, again there is no discussion on how this re-calculation occurs or indeed, how “the amount of memory required at the occurrence of an error” is ascertainable. Before the memory “error” occurred, the memory that was allocated is that which was specified in the static information that is

<sup>2</sup> Kogure, Col. 3, lines 35-38.

<sup>3</sup> Kogure, Col. 3, lines 44-46.

<sup>4</sup> Kogure, Col. 4, lines 1-6.

<sup>5</sup> Kogure, Col. 4, lines 50-51.

<sup>6</sup> Kogure, Col. 4, lines 57-59.

<sup>7</sup> Kogure, Col. 3, lines 48-52.

<sup>8</sup> Kogure, Col. 5, lines 12-14.

<sup>9</sup> Kogure, Col. 5, lines 52-55.

Appl. No. 09/666,629

Request For Reconsideration dated August 4, 2004

Reply to Final Office action of June 4, 2004

earlier read by the loader 14 and allocated by the memory management section 15. However, after the error, there is no indication of what additional mechanism specifies what additional memory is needed for the dynamic allocation.

In contrast to the preceding, claim 1 recites, among other things, "providing a memory interface within the algorithm module that supports both design-time object instantiation and dynamic object instantiation; . . . wherein the dynamic object instantiation comprises memory allocation by any framework in the plurality of frameworks *in response to memory usage requirements reported to the framework by the algorithm module through the memory interface.*" Antecedent support for this aspect exists in many places in the Specification. For example, at page 37, lines 21-24, the Specification describes that a framework can call a function "algAlloc()" of an algorithm, and that function "returns a table of memory records that describe the size, alignment, type and memory space of all buffers required by an algorithm instance (including the instance object itself)." On page 31, lines 25-26 also state, ". . . it is important to give the framework as much control over memory management as possible" and indicates that the function algAlloc() "allow[s] the algorithm module to communicate its memory usage requirements to the framework. . ." Further, the Specification describes, at pages 59-60, the dynamic instantiation and reiterates that "each algorithm has a memory interface that responds to a memory allocation inquiry (or query) with the memory usage requirements of an algorithm instance."<sup>10</sup> Moreover, the framework queries this interface to get those memory usage requirements and the algorithm responds with them, and then the framework (or "client") "allocates physical memory for [the] algorithm instance . . . based on the memory usage requirements it received."<sup>11</sup> Thus, unlike the Kogure patent which does not explain the mechanism by which it is informed of the memory aspects needed for dynamic allocation, the present invention, as set forth in claim 1, provides that memory information so that "allocation by any framework in the plurality of frameworks" may occur *"in response to memory usage requirements reported to the framework by the algorithm module through the memory interface."*

In view of the preceding, Applicants respectfully submit that the operation of the framework in claim 1 is neither shown, taught, nor suggested by the Kogure or other cited references. Accordingly, Applicants respectfully submit that claim 1 and its dependent claims 2 through 6 are in condition for allowance. In addition, independent claim 7 includes a similar recitation to that emphasized above and, thus, it and its dependent claims 8 through 11 are also in condition for allowance.

In addition to the preceding, Applicants thank the Examiner for his consideration of the remarks submitted by Applicants on March 5, 2004, as well as the Examiner's clarification with respect to those remarks. In this regard, the Examiner contends the combination of three references with respect to claims 1 and 7 is not suggested not in any of the references themselves, but instead each provides aspects of "programming

<sup>10</sup> Specification, (page 59, lines 16-17).

<sup>11</sup> Specification, (page 59, lines 23-28).

Appl. No. 09/666,629

Request For Reconsideration dated August 4, 2004

Reply to Final Office action of June 4, 2004

optimizations.” Applicants respectfully submit that such an approach is merely stating that each aspect is beneficial by itself, but does not suggest a combining with any one, or in the case of claim 1 and 7, with several other, aspects that also are beneficial for reasons pertaining to the present claims. In other words, the Examiner, having studied the present application, now knows of its benefits and is collecting various pieces from numerous different references to achieve those benefits under the broad umbrella of improving “programming optimization.” Indeed, an actual motivation of various aspects recited in the claims is explained in detail in the Background as well as other locations of the Specification. Specifically, the Specification demonstrates that there is a long history of the complexity in the digital signal processor (“DSP”) industry in that technology advances quickly, yet the time to develop the programming applications often spans years of intensive research and development due to various considerations such as “different memory management policies and I/O handling mechanisms,” and a “lack of consistent integration or programming standards.”<sup>12</sup> Other complexities include the incompatibilities of differing frameworks for differing DSPs.<sup>13</sup> Thus, the preferred embodiments, as recited in the pending claims, are not merely “programming optimizations” as stated by the Examiner, but instead are newly-combined aspects that overcome the hurdles of the multiple framework (e.g., multiple DSP) industry. These combinations provide various benefits in this particular context, and that context is reflected in the language of the pending claims (e.g., “an algorithm module that can be used without change in a plurality of frameworks”). Thus, Applicants contend that it is not proper to combine the references as has been suggested by the Examiner with hindsight to the Specification, where those references have no suggestion of combination with one another. To do otherwise is tantamount to argue that any mechanical apparatus would be obvious, after reading a specification describing it, by then combining all known mechanical elements taken to construct it because each of those elements was previously known to provide a rigid structure in a mechanical apparatus. Certainly such a benefit may be obtained from each such element, but it does not suggest a *specific* combination of those elements without first having knowledge of the specification of the overall combined apparatus. Indeed, this is even more evident as the Examiner combines an even greater number of references, such as four references with respect to claim 6, six references with respect to claims 2 and 8, seven references with respect to claims 3, 5, 9 and 10, and eight references with respect to claims 4 and 10.

Thus, in addition to or in lieu of the reasons set forth earlier with respect to the Kogure patent, Applicants respectfully submit that independent claim 1 and its dependent claims 2 through 6 and independent claim 7 and its dependent claims 8 through 11 are all in condition for allowance.

---

<sup>12</sup> Specification, page 1, line 29-page 2, line 1.

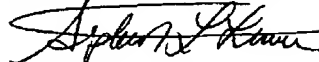
<sup>13</sup> Specification, page 3.

Appl. No. 09/666,629  
Request For Reconsideration dated August 4, 2004  
Reply to Final Office action of June 4, 2004

Conclusion

Applicants respectfully request that a timely Notice of Allowance be issued in this case.

Respectfully submitted,

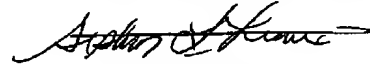


Stephen L. Levine  
Registry No. 33,413

Anderson, Levine & Lintel, L.L.P.  
14785 Preston Road, Suite 650  
Dallas, Texas 75254  
(972) 664-9552  
August 4, 2004

**CERTIFICATE OF FAX TRANSMISSION**  
**37 C.F.R. 1.8**

The undersigned hereby certifies that this correspondence is being transmitted via facsimile, on August 4, 2004, to the Patent and Trademark Office centralized facsimile number of (703) 872-9306.



Stephen L. Levine  
Registry No. 33,413